REMARKS

[0002] Applicant respectfully requests reconsideration and allowance of all of the claims of the application. Claims 1-27, 29 & 30 are presently pending. No claims are amended, withdrawn, cancelled or added herein.

Substantive Matters

Claim Rejections under § 103

[0003] The Examiner rejects claims 1-27, 29 and 30 under § 103(a). For the reasons set forth below, the Examiner has not made a prima facie case showing that the rejected claims are obvious.

[0004] Accordingly, Applicant respectfully requests that the § 103 rejections be withdrawn and the case be passed along to issuance.

[0005] The Examiner's rejections are based upon the following references alone or in combination:

- **Russell:** *Russell, et al.,* US Patent Application Publication No. 2004/0039964 (Published February 26, 2004);
- **Chinnici:** *Chinnici, et al.,* US Patent Application Publication No. 2003/0191803 (Published October 9, 2003);
- **Ernst:** *Ernst, et al.,* US Patent Application Publication No. 2003/0182308 (Published September 25, 2003); and
- **Albornoz:** *Albornoz, et al.,* US Patent Application Publication No. 2005/0154978 (Published July 14, 2005).

Overview of the Application

[0006] The Application describes a technology for securely transferring computer-readable objects across a remote boundary. The method decomposes any type of object into a hierarchy of sub-components based on a list of known

object types. Each sub-component either corresponds to a known object type or an unknown object type.

[0007] The unknown object types may be decomposed further into known object types at another level in the hierarchy. The known objects in the hierarchy are serialized into a package that is transmitted to a remote entity, which reconstructs the hierarchy. For any of the known object types, the remote entity instantiates an object and populates the object with information transmitted in the package. The decomposition may be limited by specifying a level for the hierarchy, specifying a number that limits the known objects that are serialized or specifying the properties within the object to serialized.

Cited References

[0008] The Examiner cites Russell as the primary reference in the obviousness-based rejections. The Examiner cites Chinnici, Ernst and Albornoz as secondary references in the obviousness-based rejections.

Russell

[0009] Russell describes a technology for programmatically serializing complex objects (e.g., Java Beans™). The technology programmatically generates data type mapping, responsive to encountering run-time exceptions during the serialization process. The serialization process is therefore "self-healing."

Chinnici

[0010] Chinnici describes a technology for providing an extensible serialization framework for an XML based RPC computing environment. The technology enables a computing system to receive a serialized message including a target object that is associated with at least one member object.

Ernst

[0011] Ernst describes a schema-oriented content management system for storing and accessing data and which allows content schema evolution while maintaining operation based on already stored content data.

Albornoz

[0012] Albornoz describes a technology for allowing efficient creation of data structures that correspond to data formats specified by content models specified within XML schemas. Programs written in dynamic programming language, such as JavaScript, create and instantiate object classes that conform to one or more pre-existing XML schemas. These object classes provide an application program interface (API) for application programs to manipulate data via exposed data structures and methods. Application programs are able to access exposed data structures through conventional programming methods. After the application program has completed manipulation of data within the instantiated data classes, the data is then produced as an XML document that conforms to the XML schema.

Obviousness Rejections

Lack of *Prima Facie* Case of Obviousness (MPEP § 2142)

[0013] Applicant disagrees with the Examiner's obviousness rejections. Arguments presented herein point to various aspects of the record to demonstrate that all of the criteria set forth for making a prima facie case have not been met.

Based upon Russell in view of Chinnici

[0014] The Examiner rejects claims 1-8, 11, 13, 14, 16, 17, 26, 27, 29-30 under 35 U.S.C. § 103(a) as being unpatentable over Russell in view of Chinnici. Applicant respectfully traverses the rejection of these claims and asks the Examiner to withdraw the rejection of these claims.

Independent Claims 1 and 26

[0015] Applicant submits that the combination of Russell and Chinnici does not teach or suggest at least the following features as recited in these claims (with emphasis added):

- "**negotiating with a remote entity** to determine ***which object types are known*** by the remote entity in order to determine **a list of known objects**"
- "after the negotiating, **decomposing** an object of the computer-readable objects into **multiple sub-components**, including dividing

the multiple sub-components into a **hierarchy *based upon the negotiated list of known object types***, the known object types being a type ***known by the remote entity***'

[0016] The Examiner indicates (Action, p. 3-4) the following with regard to these claims:

> As to claim, Russell teaches at least one computer-readable storage medium [p. 8, paragraph 0096] having computer executable instructions that provide a method for transferring computer-readable objects [p. 5, paragraph 0071] across a remote boundary [p. 7, paragraph 0088], the method comprising:
>
> decomposing an object of the computer-readable objects [complex object 500 that needs to be serialized; p. 5, paragraph 0063] into multiple sub-components [A child element 930 is generated for the top-level class of the object being serialized; p. 6, paragraph 0080] including dividing the multiple sub-components into a hierarchy based upon the negotiated list of known object types [A child element 930 is generated for the top-level class of the object being serialized; p. 6, paragraph 00801, the known object types being a type known by the remote entity [type mappings; p. paragraph 0077];
>
> serializing the multiple sub-components into a serialized package [entire complex object has been marshalled, the result is a serialized object as shown at 540; p. 4, paragraph 0057]; and
>
> transmitting the serialized package to the remote entity [Message 1015 is then sent through the network 1020,1025, and then reaches the target Web service 1030; p. 7, paragraphs 0087 and 0088 and p. 3, paragraph 0043]. Russell does not teach negotiating with a remote entity to determine which object types are known by the remote entity in order to determine a list of known objects.
>
> However, Chinnici teaches negotiating with a remote entity to determine which object types are known by the remote entity in order to determine a list of known objects [paragraphs 0119, 0133, 0134, 0137, and 0144], after the negotiating, decomposing an object of the computer-readable objects into multiple sub-components [paragraph 0154 and 0174].

[0017] The presently claimed negotiation process, unlike existing systems, not only provides a mechanism for supporting communication between **two different versions**, but also minimizes the amount of data transferred over the remote boundary. See, for example, Specification at page 5 lines 1-7. In

particular, the protocol negotiation process allows both the requesting entity and the remote entity to **upgrade their system *independent of each other*** while still allowing them to communicate with each other **using *different* versions**. See, for example, Specification at page 36, lines 1-10. In practice, this allows a requesting entity, which is operating with a version of base types that is several years newer than the version on the remote entity, the ability to still communicate with the remote entity. The converse is also true. Therefore, systems that have been shipped many years ago can be managed by newer administrator computers.

[0018] The Examiner admits (Action, p. 4) that Russell fails to teach "negotiating with a remote entity to determine which object types are known by the remote entity in order to determine **a list of known objects.**" Therefore, contrary to what the Examiner asserts (Action, p. 4), Russell also fails to teach or disclose "dividing the multiple sub-components into a hierarchy **based upon a *negotiated* list of known object types**." Russell at most describes serializing an object **based on a mapping** (known to the client) **that is dynamically generated while serializing**, not **negotiated** with a remote entity. See, for example, Russell at paragraphs [0074], [0077] and [0080]. The programmatically-generated mappings are **automatically added to a mapping registry accessible to the client**, such that references to the corresponding object during a subsequent serialization process will no longer generate an exception. See, for example, Russell at paragraphs [0035]-[0037] and [0087]. In other words, Russell describes changing and updating the current "version" of the client with the programmatically generated type

mappings. Nowhere in Russell makes any reference as to what is "**known**" to a **remote entity**, nor does any **negotiation** with a remote entity occur.

[0019] The Examiner relies on Chinnici to compensate for the defects of Russell. Applicant respectfully disagrees. Like Russell, Chinnici fails to teach or disclose "negotiating with a remote entity to determine which object types are known by the remote entity in order to determine **a list of known objects"** and "dividing the multiple sub-components into a ***hierarchy*** **based upon a** ***negotiated*** **list of known object types**." Although Chinnici describes mapping of Java types and XML data types in the cited paragraphs [0119], [0133], [0134], [0137] and [0144], it is unclear what the Examiner is equating to the "**negotiated** list of known object types." In fact, there is no motivation to **negotiate with a remote entity** since Chinnici describes a serializer that is defined for Java arrays and classes in the ***standard*** **Java Collection framework**. See, for example, Chinnici at paragraph [0129].

[0020] As shown above, the combination of Russell and Chinnici does not teach or suggest all of the elements and features of these claims. Also, there is no reason to combine the teachings of the references. Accordingly, Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 2-8, 11, 13,14, 16, 17, 27, 29-30

[0021] These claims ultimately depend upon independent claims 1 and 26. As discussed above, claims 1 and 26 are allowable. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Additionally, some or all of these claims may also be allowable for additional independent reasons.

Based upon Russell in view of Chinnici and Ernst

[0022] The Examiner rejects claims 9, 10 and 12 under 35 U.S.C. § 103(a) as being unpatentable over Russell in view of Chinnici and further in view of Ernst. Applicant respectfully traverses the rejection of these claims and asks the Examiner to withdraw the rejection of these claims.

[0023] Ernst is only cited as teaching the additional recitations of claims 9, 10 and 12 and does not cure the deficiencies of Russell and Chinnici, with regard to independent claim 1 as discussed above. Claims 9, 10 and 12 ultimately depend upon claim 1. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Thus, at least because of their dependence on claim 1, claims 9, 10 and 12 are patentable over Russell, Chinnici and Ernst. Additionally, some or all of these claims may also be allowable for additional independent reasons.

Based upon Russell in view of Chinnici and Albornoz

[0024] The Examiner rejects claims 15 and 18-25 under 35 U.S.C. § 103(a) as being unpatentable over Russell in view of Chinnici and further in view of Albornoz. Applicant respectfully traverses the rejection of these claims and asks the Examiner to withdraw the rejection of these claims.

[0025] Applicant includes herein a §131 affidavit to remove Albornoz as a reference. Applicant respectfully requests the Examiner to consider the affidavit and, if some defect in it be found, inform the Applicant of the defect so that the Applicant may correct the defect.

[0026] But regardless of the affidavit, Albornoz is only cited as teaching the additional recitations of claims 15 and 18 and does not cure the deficiencies of Russell and Chinnici, with regard to independent claim 1 as discussed above. Claims 15 and 18 ultimately depend upon claim 1. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Thus, at least because of their dependence on claim 1, claims 15 and 18 are patentable over Russell, Chinnici and Albornoz. Additionally, some or all of these claims may also be allowable for additional independent reasons.

[0027] In addition, independent claim 19 includes recitations similar to those of claim 1, making specific reference to "the list having been ***negotiated*** **with the remote entity** before the serializing of the package on the remote entity." As discussed above, neither Russell nor Chinnici, alone or in combination, teach or suggest a **remote entity** or a list ***negotiated*** prior to serializing. Since Albornoz fails to compensate for the defect, the combination of

Russell, Chinnici and Albornoz fails to teach or suggest the features of independent claim 19.

[0028] Claims 20-25 ultimately depend upon claim 19. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Thus, at least because of their dependence on claim 19, claims 20-25 are patentable over Russell, Chinnici and Albornoz. Additionally, some or all of these claims may also be allowable for additional independent reasons.

Conclusion

[0029] All pending claims are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the application. If any issues remain that prevent issuance of this application, the **Examiner is urged to contact me before issuing a subsequent Action**. Please call or email me at your convenience.

Respectfully Submitted,

Lee & Hayes, PLLC
Representatives for Applicant

/kaseychristie40559/ Dated: 2/11/2009

Kasey C. Christie (kasey@leehayes.com; 509-944-4732)
Registration No. 40559
Customer No. **22801**

Telephone: (509) 324-9256
Facsimile: (509) 323-8979
www.leehayes.com